Cooperative coevolutionary instance selection for multilabel problems[⋆]

Nicolás García-Pedrajas[a,∗], Gonzalo Cerruela-García[a]

[a] *University of Córdoba, Campus de Rabanales, 14011 Córdoba (Spain)*

Abstract

Multilabel classification as a data mining task has recently attracted greater research interest. Many current data mining applications address problems having instances that belong to more than one class, which requires the development of new efficient methods. Advantageously using the correlation among different labels can provide better performance than methods that deal with each label separately.

Instance-based classification models, such as k-nearest neighbors for multilabel datasets, ML-kNN, are among the best performing methods on any classification task and have also been successfully applied to multilabel problems. Despite their simplicity, they achieve comparable performance to considerably more complex methods. One of the challenges associated with instance-based classification models is their requirement for storing all the training instances in memory. To ameliorate this problem, instance selection methods have been proposed. However, their application to multilabel problems is problematic because the adaptation of most of their concepts to multilabel problems is difficult.

In this paper, we propose a cooperative coevolutionary algorithm for instance selection for multilabel problems. Two different populations evolve together cooperatively. One of the populations is devoted to obtaining solu-

[⋆]This work was supported in part by grant PID2019-109481GB-I00/AEI/q10.13039/501100011033 of the Spanish Ministry of Science and Innovation and grant UCO-1264182 of the Junta de Andalucía Excellence in Research Program and FEDER Funds.

[∗]Corresponding author

Email addresses: `npedrajas@uco.es` (Nicolás García-Pedrajas), `gcerruela@uco.es` (Gonzalo Cerruela-García)

tions for each label, whereas the other population combines these results into solutions for the instance selection for multilabel dataset tasks. On a large set of 70 real-world problems, our approach improves the results of both the ML-kNN method with the whole dataset and an instance selection method using a standard evolutionary algorithm.

Keywords: Multilabel learning, instance selection, cooperative coevolution, instance-based learning.

---

1. Introduction

Many modern applications involve vast amounts of data for classification and increasingly complex categorization schemes where one instance of data may simultaneously belong to several topics. This task is typically termed multilabel learning [1]. In contrast with single label classification, where each instance is associated with only one class, multilabel classification is concerned with learning from instances where each instance can be associated with multiple labels. The generality of multilabel problems makes them more difficult to solve than their single label binary or multiclass counterparts.

Multilabel learning is a special case of the more general multioutput learning [2]. In multioutput learning, we have a set of discrete labels that are not necessarily binary, while in multilabel learning, all labels have only binary values. Although this paper is devoted to multilabel problems, the proposed work could be easily extended to multioutput data. As a result of the growing interest multilabel classification has received over the past few years, a variety of multilabel learning methods have been developed and applied to diverse problems, including the following: text categorization, the automatic annotation of multimedia contents, web mining, rule mining, cheminformatics, bioinformatics, information retrieval, etc.

Formally, we can define a multilabel problem as follows [1]: Let T be a multilabel evaluation dataset consisting of p multilabel instances $\mathbf{x}_i$ and their associated label set y_i, where $T = \{(\mathbf{x}_i, y_i)\}, 1 \leq i \leq p$, $(\mathbf{x}_i \in X, y_i \in \mathcal{Y} = \{0,1\}^q)$ with a label set L, where $|L| = q$. Let h be a multilabel classifier and $h(\mathbf{x}_i) = \{0,1\}^q$ be the set of labels predicted by h for the instance $\mathbf{x}_i$. Let $f(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in X$, $y_i \in \mathcal{Y}$ be a real-valued function $f : X \times \mathcal{Y} \to R$. A successful learning system would tend to output larger values for function f for the labels in y_i versus those not in y_i. The real-valued function f can be easily transformed to a ranking function, $rank_f(\mathbf{x}_i, y_i)$, where $rank_f$ is the predicted rank of label y_i for instance $\mathbf{x}_i$. $h(\mathbf{x}_i)$ can be obtained from $f(\mathbf{x}_i)$ when an appropriate threshold is set.

The key challenge of multilabel learning is taking advantage of the correlations among labels to address the exponential growth of the label space with the number of distinct labels, 2^q, for q different labels. Methods that deal with multilabel datasets without considering the relationships between labels are just solving a group of independent binary problems. There are two broad categories of methods to deal with multilabel problems [1]: problem transformation methods and algorithm adaptation methods. Problem transformation methods tackle the multilabel learning problem by transforming it into other well-established learning scenarios. Algorithm adaptation methods tackle the multilabel learning problem by adapting well known learning techniques to deal with multilabel data directly. Among the best performing methods in multilabel learning is the adaptation of the k-nearest neighbors to multilabel datasets, the Multilabel k-Nearest Neighbors (ML-kNN) [3] method. However, as in the single label case, the method has the drawback of needing all the training set stored in memory, thus increasing the runtime of the algorithm and increasing the needed resources. Instance selection is a way to address this problem and improve the performance of instance-based classifiers [4].

Cooperative coevolution [5, 6] is a paradigm in the area of evolutionary computation based on the evolution of coadapted subcomponents without external interaction. In cooperative coevolution, a number of species evolve together. Cooperation among individuals is encouraged by rewarding the individuals for their joint efforts to solve a target problem. The work in this area has shown that cooperative coevolutionary models present many interesting features, such as specialization through genetic isolation, generalization and efficiency [7]. Cooperative coevolution approaches the design of modular systems in a natural way since the modularity is part of the model. Other models need some *a priori* knowledge to decompose the problem *by hand*. In many cases, either this knowledge is not available or it is not clear how to decompose the problem.

The advantage of cooperative coevolution over a traditional evolutionary algorithm is mainly because of its divide-and-conquer decomposition strategy [6]. Ma *et al.* [6] identified four main advantages: i) the decomposition of the problem allows parallelism to speed up the optimization process; ii) each subproblem is solved with a separate subpopulation, which maintains good solution diversity; iii) decomposing a system into submodules increases the robustness against the modules' errors and failures and thus enhances the reusability in dynamic environments; and iv) in the field of function optimization, it reduces the "curse of dimensionality". Cooperative coevolution has been successfully adapted to optimization tasks both in standard single objective optimization [8] and multiobjective optimization [9]. Its ability to decompose a difficult problem into smaller easier problems is very useful for complex optimization problems with many variables. This decomposition ability can also be of great help in multilabel learning. Cooperative coevolution can also be coupled with memetic algorithms [10] and culture-based learning [11] to deal with the complex problems posed by multilabel classification.

In this paper, we propose a cooperative coevolutionary model applied to instance selection for multilabel datasets. The proposed model combines a local and a global approach that cooperate via a coevolutionary model. This model is implemented using two separate populations. A low level population is divided into subpopulations, each of which is devoted to exclusively one label. A higher level population combines individuals from this low level population to obtain a selection of instances for the global multilabel problem. The locality of cooperative coevolution allows the exploration of higher-order correlations without exponentially increasing the computational complexity. Furthermore, the inherent distributive nature of cooperative coevolution offers a flexible approach that can handle complex problems with a feasible computational cost.

The model allows the decomposition of the multilabel problem in simpler single label problems while keeping the global approach through the population of combinations of subindividuals. The methods can be efficiently implemented since much of the evolution can be parallelized. To the best of our knowledge, this is the first cooperative approach to instance selection using instance-based learners for multilabel problems.

The remainder of this paper is organized as follows: Section 2 describes the scarce existing related work; Section 3 describes our proposal in depth; Section 4 explains the experimental setup; Section 5 shows the experimental results and their discussion and, finally, Section 6 summarizes the conclusions of our work.

2. Related work

Very few works have been devoted to instance selection for multilabel problems. Arnáiz-González *et al.* [12] extended the concept of local sets used for single label instance selection to the multilabel case [4]. The developed method achieved good performance in a set of 11 problems. Kordos *et al.* [13] developed an evolutionary method for a multioutput regression.

Cooperative coevolution has been extensively used for large scale optimization [14, 15, 16, 17]. However, there are very few attempts that use the properties of cooperative coevolution for multilabel problems. Sun *et al.* [18] proposed a coevolutionary multilabel hypernetwork to address the learning of higher-order relationships among labels. Rosales-Pérez *et al.* [19] proposed a cooperative coevolutionary method for hyperheuristic design for multilabel problems. Park *et al.* [20] developed a multipopulation genetic algorithm for multilabel feature selection. Although there is no cooperative approach, the use of multiple populations shares part of the philosophy of our proposal.

3. Cooperative coevolution for instance selection in multilabel datasets

In cooperative coevolution [7], instead of evolving individuals to solve the whole task, subcomponents that represent a partial solution are evolved. These subcomponents may be evolved independently. Then, collaboration among the subcomponents must be established to obtain whole solutions. There may be various types of collaboration, such as combining the best subcomponent from different subpopulations or using a population of combinations [21]. Cooperative coevolution has been proved to be efficient at dealing with large scale optimization problems [22] and Big Data [10]. Furthermore, the possibility of running concurrently [23] is a major advantage for large and enormous problems.

The original cooperative coevolution framework can be generally defined as follows [24]:

1. Problem decomposition: Decompose a complex problem into smaller subcomponents that can be solved using evolutionary computations. These subcomponents can be heterogeneous and are evolved using subpopulations.
2. Subcomponent optimization: Evolve each subpopulation using a suitable evolutionary computation algorithm. Usually, subpopulations are evolved independently.

3. Fitness evaluation: The fitness of the subcomponents is evaluated cooperatively.

Additionally, a second level of evolution where subcomponents are combined using another population that is evolved concurrently can be implemented [25]. In this case, we have a global and a local evolution that evolve together. The global evolution represents the solution to the problem and the local evolution represents the partial solution to different aspects. In that case, there is also local and global interrelated fitness values [26]. This is the model used in this paper.

The key of cooperative coevolution is decomposing the optimization problem into smaller subproblems that can be independently addressed. Each subpopulation seeks the optimal solutions for a subproblem in the corresponding search space [9]. A complete solution can be obtained by combining the best individuals of every subpopulation or by means of another population of combinations [21]. Decomposing the problem into subproblems is one of the most important tasks in the design of a cooperative algorithm. For the multilabel problem, an intuitive and straightforward approach is considering only one of the labels for each subpopulation. This is the basic approach that we use in this paper. Cooperation mechanisms are provided to account for the higher level relationships among the labels.

The success of any cooperative algorithm depends on two key aspects: the problem decomposition and collaboration. First, the subindividual must be designed to solve meaningful subproblems; and second, they must collaborate effectively to solve the overall problem. Usually, collaboration is carried out using one of two strategies [27]: collaborator selection pressure, i.e., the degree of greediness in choosing a representative member from a subpopulation to form the complete solution(s) of the original problem; or the collaborator pool size, i.e., the numbers of the complete problem solutions used to evaluate the fitness of an individual in a particular subpopulation.

Our cooperative model is based on the use of two separate populations that evolve cooperatively: a global population P_{ML} whose target is the multilabel problem; and a local population P_{SL} that is made up of as many subpopulations, SP_i, as labels in the dataset, $q = |L|$. The subpopulation $SP_i \in P_{SL}$ evolves considering only label i using a CHC evolutionary algorithm. Since the different subpopulations evolve independently, a parallel implementation is possible.

The global population is used to obtain the global selection, which is

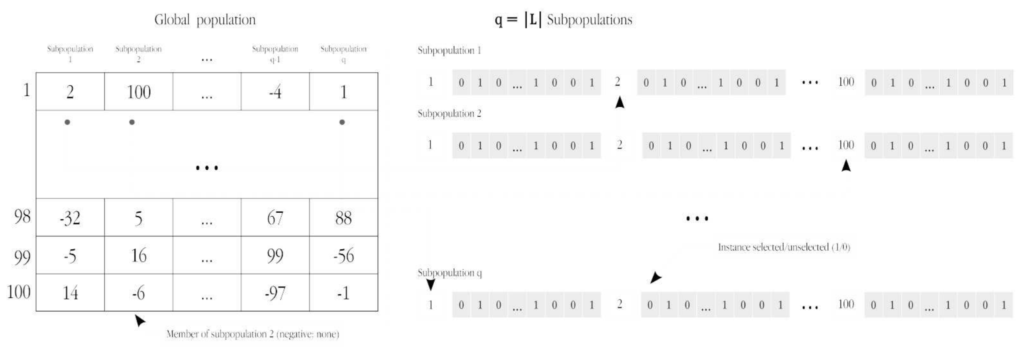

Figure 1: Global population and populations of subindividuals. Each element of the global population is a reference to an individual of the corresponding subpopulation of subindividuals that carries out an instance selection process in its corresponding label.

the final outcome of the algorithm. The individuals of this population are arrays of q integers that select one individual from every subpopulation. The method for combining the subindividuals to obtain a selection of instances is explained below. The individuals of the population are integers in the range $[-N, N]$, where N is the size of the subpopulations. Dynamic grouping has been proposed as a way of better modeling the interactions among variables if cooperative coevolutionary optimization is used [28]. We use a dynamic allocation of resources allowing the individuals to have negative values, which means that no individual is selected from the corresponding subpopulation. A general view of the architecture of the evolutionary process is depicted in Figure 1. The two populations evolve sequentially. Each generation of the whole system consists of N_p generations of the global population followed by N_{sp} generations of the population of subpopulations.

Each member, $P_i \in P_{\text{ML}}$, of the global population is composed of three parts that are evolved together, $P_i = (\mathbf{p}_i, qs_i, k_i)$. $\mathbf{p}_i$ is the array considering the members of each subpopulation. The other two values are used to improve the ability of the algorithm to find useful solutions. k_i is used as the number of neighbors considered in the ML-kNN, which is evolved alongside the selection. qs_i is the threshold used to select a certain instance and it is explained in Section 3.1.

The evolution of this global population is carried out using an adaptation of the CHC genetic algorithm [29] to individuals represented by integers. The population of size S_p is evolved for a fixed number of generations. In each generation, all the individuals are randomly matched in pairs to produce a population of size $2 \times S_p$. The crossover operator is based on the HUX operator. To match the two $\mathbf{p}_i$ parts of the individual, each offspring inherits one of the parents in turn. The k_i and qs_i parts are matched using a BLX-α crossover. After matching, the best N_p individuals are chosen to survive. Mutation is not used within the standard CHC algorithm, however in order to improve the performance we use three different mutation operators, one of them including a local search algorithm. These three operators are:

1. *Local search mutation.* This operator is a local search used to improve the fitness of the mutated individual. For the individual P_i, using a random order, all the members of $\mathbf{p}_i$ flip their sign. This means that if p_{ij} was positive, the mutated individual does not consider any member of subpopulation j; and if it was negative, then a member from population j is added to P_i. After each modification, the individual is reevaluated and the mutation is kept if its fitness is improved. This operator is applied with a probability p_{ls}.
2. *Bit mutation.* This operator works as a standard random bit mutation, but it affects all subindividuals that are members of the mutated individual. First, the selection represented by the individual is obtained using the method explained in Section 3.1. Then, a bit mutation is applied with a probability p_{bit} to a randomly selected bit. There are two differences with standard bit mutation: the mutation is only kept if the fitness of the individual is improved, and the mutation of the bit is inherited by all the subindividuals that participate in the mutated individual. This mutation is applied with probability p_{bm}.
3. *Random mutation.* This mutation modifies a randomly selected individual, which is assigned a new member of the corresponding subpopulation. This mutation is applied with probability p_{rm}.

To avoid removing the best individuals, we also use elitism. The best p_{elite} percentage of the population is not subject to mutation. After the population has been evolved for N_p iterations, all the subpopulations are evolved for N_{sp} iterations. Each subpopulation evolves independently also following a CHC algorithm. Since each element of a subpopulation is a binary array, we have implemented an standard CHC algorithm.

However, in order to improve the results, we also use local search at the subpopulation level implemented as a mutation operator. In single label instance selection, memetic algorithms have been proven to be superior to other types of genetic algorithms in previous works [30]. However, for the multilabel case, it is not easy to implement a memetic algorithm since there are very few methods that can be implemented as local optimization operators. However, for our cooperative approach, the implementation of a memetic algorithm is straightforward at the subpopulation level.

Within the CHC algorithm carried out for every subpopulation, we introduce a local search that consisted of a single label instance selection algorithm. Since most of the labels are class-imbalanced, if they are considered in isolation, we use two different local search algorithms: a class-imbalanced instance selection algorithm, the one-sided selection (OSS) algorithm [31]; and a standard instance selection algorithm, the Drop3 algorithm [32]. Both algorithms are carried out independently with probabilities p_{oss} and p_{drop3} for the OSS and Drop3 methods, respectively.

The last concept introduced in our evolution is *elitism inheritance*. In the global population, we use the elitism mentioned above. However, since the global population is made of combinations of the members of the subpopulations, a modification of one of these members would affect the global individual and might damage its performance. To avoid this, the elite consideration is inherited by all members of an individual. That means that if a certain individual is among the top p_{elite} individuals, all its members are also considered elite individuals and are kept during the evolution of the subpopulations. This is what we call elitism inheritance. The elitism in the subpopulations is limited by p_{elite} since, at most, this percentage of subindividuals could belong to an elite member of the global population.

To initialize the individuals of the population, each value is selected randomly in the range $[-N, N]$, which means that initially each individual has members of all the subpopulations. The subindividuals are initialized randomly with a probability of having an instance selected of p_{init}.

The complete cooperative coevolutionary algorithm is shown in Algorithm 1.

3.1. Fitness of individuals and subindividuals

The evaluation of the individuals of the global population involves two steps. First, we must obtain a selection from the individual. Since each member of a subpopulation represents a complete selection, we must devise a way of combining the selections carried out by every subindividual. The most straightforward way is majority voting. An instance is selected if at least half of the members of the subpopulations selected it. However, from

Algorithm 1: Cooperative Coevolutionary Instance Selection for Multilabel problems (CCISML).

```
   Data: A training set $T = \{(\mathbf{x}_i, y_i)\}, 1 \leq i \leq p$, $(\mathbf{x}_i \in X, y_i \in \mathcal{Y} = \{0,1\}^q)$, with a label set $L$, $|L| = q$
   Result: The subset of selected instances $S \subset T$.
2  Initialize global population
4  Initialize subpopulations
5  for $g = 1$ to $G$ do
       /* Evolve population                                          */
6      for $i = 1 to N_p$ do
8          Perform crossover of all individuals
10         Perform local search mutation with probability $p_{ls}$
12         Perform bit mutation with probability $p_{bit}$
14         Perform random bit mutation for nonelite individuals with probability $p_{rm}$
16         Revaluate individuals
18         Obtain best individuals to survive to next iteration
       /* Evolve subpopulations                                      */
19     for $i = 1 to N_{sp}$ do
20         foreach Subpopulation do
22             Perform CHC crossover of all individuals
24             Perform Drop3 mutation with probability $p_{drop3}$ for nonelite subindividuals
26             Perform OSS mutation with probability $p_{oss}$ for nonelite subindividuals
28             Revaluate individuals
30             Obtain the best individuals to survive to the next iteration
32     If time exceeded break
34 Return $S$ as the selection performed by the best individual
```

a practical point of view, this method achieved poor results. Furthermore, depending on the number of actual members of the individual, which can be different since negative values are not considered, the use of a fixed number as a threshold can be troublesome. Thus, we opted for the *quorum sensing* method [33]. Each subpopulation adds a vote to the corresponding instance; and when the votes are above a sensing threshold, the instance is selected. The threshold is dynamic, and it evolves together with the remaining part of the individual, as we explained above.

To evaluate a certain individual $P_i = (\mathbf{p}_i, qs_i, k_i)$, the selection of instances $S_i \subset T$ is obtained by combining the member of the subpopulations selected by $\mathbf{p}_i$ and the threshold value given by qs_i. Then, an MLkNN algorithm is applied to S_i using k_i as the number of nearest neighbors to obtain the classification performance of the subset $C_i(S_i, k_i)$. The reduction is measured as the percentage of removed instances $R_i = 1 - \frac{|S_i|}{|T|}$. As it is usual in evolutionary instance selection for single label problems [34], the fitness value is a weighted combination of these two values. Another term was added to the fitness function to improve the results for both our approach and the standard CHC algorithm, the absolute value of the difference of the average number of predicted labels, n_p, and the average number of relevant labels, n_r. The fitness function is:

$$F_i = C_i(S_i, k_i) + R_i - |n_p - n_r|/q. \tag{1}$$

Classification performance is more difficult to evaluate in multilabel datasets since there are many metrics that consider the accuracy of a classifier from different points of view. These metrics are explained in Section 4.1. The experiments provide results using each of these metrics separately to study the behavior of our proposal when the classification performance is aimed at different targets. This same fitness function was used for the standard CHC genetic algorithm used to compare the performance of our cooperative proposal with others.

The evaluation of the members of the subpopulations is more complicated. Usually, in cooperative coevolution, the fitness of an individual in a subpopulation is evaluated in terms of the complete solution(s) in which this individual participates [6]. However, we also considered the isolated performance of each subindividual. Thus, we consider fitness using three factors: the reduction of the subindividual, its performance for the label assigned to the subpopulation and the average fitness of the members of the global population in which it participates as a member. Thus, the evaluation of every subindividual j of the subpopulation i, s_{ij}, is given by:

$$f_{ij} = c_i(s_{ij}) + r_{ij} + \bar{F}, \tag{2}$$

where $c_i(s_{ij})$ is the classification performance of the k-nearest neighbors for label i measured using Matthews correlation coefficient (MCC) using the selection of instances given by s_{ij}:

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}}, \tag{3}$$

we use MCC because most labels present a high class-imbalance problem, r_{ij} is the reduction obtained by s_{ij}, and $\bar{F}$ is the average fitness of the members of the global population in which s_{ij} participates. In this way, the fitness of the subindividuals considered a local part, which is measured by the reduction and single label classification performance; and a global part, which is measured by the average fitness of the global individuals in which the subindividual participates.

To improve traditional cooperative coevolution, Omidvar et al. [35] proposed a contribution-based cooperative coevolution framework in which the subcomponents that have the largest contributions to the global objective value will receive extra fitness evaluations to evolve. When using this method, every subpopulation would not be treated equally since some of them are more important for the whole evolutionary process than others. However, we experimentally tested this idea and achieved worse results; thus, we did not incorporate it into our model.

4. Experimental setup

To make a fair comparison between the standard instance selection evolutionary algorithm and our cooperative coevolution-based proposal, we considered 70 datasets whose characteristics are shown in Table 1. These datasets represent a varied number of problems with different numbers of instances, features and labels. Furthermore, the label densities are also very different among the datasets.

Table 1: Description of the datasets

	Dataset	Instances	Features	Labels
1	20NG-F	19300	1006	20
2	3s-bbc1000	352	1000	6
3	3s-guardian1000	302	1000	6
4	3s-inter3000	169	3000	69
5	3s-reuters1000	294	1000	6
6	bibtex	7395	1836	1594
7	birds	645	271	193
8	CAL500	502	68	1744
9	corel16k001	13766	500	1534
10	corel16k002	13761	500	1643
11	corel16k003	13760	500	1542
12	corel16k004	13837	500	1620
13	corel16k005	13847	500	1602
14	corel16k006	13859	500	1624
15	corel16k007	13915	500	1748
16	corel16k008	13864	500	1683
17	corel16k009	13884	500	1733
18	corel16k010	13618	500	1443
19	Corel5k	5000	499	373
20	emotions	593	72	6
21	enron	1702	1001	53

Continued on next page

	Dataset	Instances	Features	Labels
22	EukaryoteGO	7766	12689	22
23	EukaryotePseAAC	7766	440	22
24	Eurlex-sm	19348	5000	201
25	flags	194	43	71
26	foodtruck	407	31	12
27	genbase	662	1185	27
28	GnegativeGO	1392	1717	8
29	GnegativePseAAC	1392	440	8
30	GpositiveGO	519	912	43
31	GpositivePseAAC	519	440	4
32	HumanGO	3106	9844	14
33	HumanPseAAC	3106	440	14
34	Image	2000	294	5
35	LLOG-F	1460	1003	75
36	medical	978	1449	45
37	Music	592	71	6
38	OHSUMED-F	13929	1002	23
39	PlantGO	978	3091	12
40	PlantPseAAC	978	440	12
41	rcv1subset1	6000	47236	101
42	rcv1subset2	6000	47236	101
43	rcv1subset3	6000	47236	101
44	rcv1subset4	6000	47236	101
45	rcv1subset5	6000	47236	101
46	REUTERS-K500-EX2	6000	500	103
47	scene	2407	294	6
48	SLASHDOT-F	3782	1079	22
49	Stackex_chemistry	6861	540	175
50	Stackex_chess	1675	585	227
51	Stackex_coffee	225	1763	123
52	Stackex_cooking	10491	577	400
53	Stackex_cs	9270	635	274
54	Stackex_philosophy	3971	842	233
55	VirusGO	207	749	6
56	VirusPseAAC	207	440	6
57	Water-quality	1060	16	14
58	Yahoo_Arts	7484	23146	26
59	Yahoo_Business	11214	21924	30
60	Yahoo_Computers	12444	34096	33
61	Yahoo_Education	12030	27534	33
62	Yahoo_Entertainment	12730	32001	21
63	Yahoo_Health	9205	30605	32
64	Yahoo_Recreation	12828	30324	22

Continued on next page

	Dataset	Instances	Features	Labels
65	Yahoo_Reference	8027	39679	33
66	Yahoo_Science	6428	37187	40
67	Yahoo_Social	12111	52350	39
68	Yahoo_Society	14512	31802	27
69	yeast	2417	103	14
70	Yelp	10806	671	5

Regarding comparisons, we used the Wilcoxon test [36] as the main statistical test for comparing pairs of algorithms. This test was chosen since it assumes limited commensurability and is safer than parametric tests since it does not assume normal distributions or homogeneous variance. Thus, this test can be applied to classification performance and reduction ability. Furthermore, empirical results [36] show that this test is stronger than other tests. Since we are testing our approach with respect to the standard voting procedure, our comparison will always be pairwise. We consider a confidence level of 95% in all the tests.

One of the problems when comparing a standard evolutionary algorithm and a cooperative one is how to fairly run the two algorithms. In most comparisons of evolutionary algorithms, the number of evaluations of the fitness function can be used as a way of comparing the methods with similar computational costs. In our case, that it is not possible since the evaluation of the individuals of the populations and subpopulations is very different in terms of the procedures and computational costs. Thus, we opted for comparing the methods using the total runtime. For both approaches, standard and coevolutionary, the algorithm was run for a maximum of one hour and the best individual after that time was returned as result of the evolution. With this setup, both methods are compared when the computation costs are the same.

Although there are several versions of ML-kNN [37, 38, 39] we used in our experiments the original version to avoid and over complicated implementation that could preclude the conclusions of our work.

The source code, which was written in C and licensed under the GNU General Public License, and the datasets are freely available upon request from the authors.

4.1. Evaluation measures

The evaluation of multilabel classification methods is relatively difficult because the prediction for an instance is a set of labels, and the result can be fully correct, partially correct (with different levels of correctness) or fully incorrect [40, 41]. Thus, many different metrics have been proposed [1]. The metrics can be divided into two major groups: *example-based* (EB) metrics and *label-based* (LB) metrics. The former evaluate the learning system on each instance (example) separately and then obtain a unique measure for the average value across the test set. The latter obtains the performance of the learning system for each class label separately and then returns a unique measure by means of macro/microaveraging across all class labels. Furthermore, the metrics can be focused on classification (using $h(\cdot)$) or ranking (using $f(\cdot)$).

There are many metrics defined in the literature [1]. Among them, we have chosen the following metrics to be used to compare the results of the studied metrics. They offer a comprehensive set of measures that evaluate the performance from different points of view.

4.1.1. Example-based metrics

For classification, the most relevant example-based metrics are given below:

1. *Subset accuracy* evaluates the fraction of correctly classified examples, that is, the examples for which the predicted set of labels is identical to the set of relevant labels:

$$\text{subsetacc}(h) = \frac{1}{p}\sum_{i=1}^{p}[\![h(\mathbf{x}_i) = y_i]\!] \tag{4}$$

   where $[\![\pi]\!]$ returns 1 if predicate π is true and 0 otherwise.
   This metric can be considered to be the multilabel counterpart of the accuracy metric. It can be overly strict, especially if the number of labels, q, is large. As it is the case for the accuracy, a perfect model would have a value of 1 for this metric.

2. *Hamming Loss* [42] evaluates the number of times a label not belonging to an instance is predicted or a label belonging to an instance is not predicted.

$$\text{Hamming Loss}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}|h(\mathbf{x}_i)\Delta y_i|, \tag{5}$$

   where Δ indicates the symmetric difference between two sets and corresponds to the XOR operation in Boolean logic. With respect to this metric, the performance is optimal when its value is zero, and higher values signify lower performance.

3. *Accuracy* [42, 43] is defined as the proportion of the number of correctly predicted labels to the total number (predicted and actual) of labels for an instance:

$$\text{Accuracy}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|h(\mathbf{x}_i)\bigcap y_i|}{|h(\mathbf{x}_i)\bigcup y_i|} \tag{6}$$

With respect to this metric, the performance is optimal when its value is one, and lower values indicate lower performance.

4. *Precision* [42, 43] is the average proportion of the number of correctly predicted labels to the total number of predicted labels:

$$\text{Precision}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|h(\mathbf{x}_i)\bigcap y_i|}{|h(\mathbf{x}_i)|} \tag{7}$$

5. *Recall* is the average proportion of the number of correctly predicted labels to the total number of relevant labels:

$$\text{Recall}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|h(\mathbf{x}_i)\bigcap y_i|}{|y_i)|} \tag{8}$$

With respect to this metric, the performance is optimal when its value is one, and lower values indicate lower performance.

6. F^β is an integrated version of the precision and recall with a balancing factor β:

$$F^\beta = \frac{(1+\beta^2)\cdot \mathit{Precision}(h)\cdot \mathit{Recall}(h)}{\beta^2 \mathit{Precision}(h) + \mathit{Recall}(h)}. \tag{9}$$

In the experiments, we show the results for $\beta = 1$.

These previous metrics are devoted to classification. We also consider the following example-based metrics used for measuring the ability of a classifier from the point of view of ranking the labels:

1. *One-error* [42] measures the number of times the most highly ranked predicted label is not included in the set of actual labels of an instance:

$$\text{One-error}(f) = \frac{1}{p}\sum_{i=1}^{p}[\![[\arg\max_{y\in\mathcal{Y}} f(\mathbf{x}_i, y)] \notin y_i]\!] \tag{10}$$

With respect to this metric, the performance is optimal when its value is zero, and higher values indicate lower performance.

2. *Coverage* [42] evaluates how far, on average, we must descend down the ranked list to cover all the proper labels of an instance:

$$\text{Coverage}(f) = \frac{1}{p}\sum_{i=1}^{p} \max_{y\in y_i} rank_f(\mathbf{x}_i, y) - 1 \tag{11}$$

   With respect to this metric, higher values indicate lower performance.

3. *Ranking Loss* [42] denotes the average fraction of label pairs that are reversely ordered for an instance and expresses the frequency of irrelevant labels being ranked higher than relevant labels:

$$\begin{aligned}&\text{Ranking Loss}(f) = \\ &\frac{1}{p}\sum_{i=1}^{p}\frac{1}{|y_i||\overline{y_i}|}|\{(y_a, y_b) : f(\mathbf{x}_i, y_a) \leq f(\mathbf{x}_i, y_b), \\ &(y_a, y_b) \in y_i \times \overline{y_i}\}|,\end{aligned} \tag{12}$$

   where $\overline{y_i}$ is the complementary set of y_i with respect to L. With respect to this metric, the performance is optimal when its value is zero, and higher values indicate lower performance.

4. *Average precision* evaluates the average fraction of relevant labels that are ranked higher than a particular label $y \in y_i$ by the model:

$$\begin{aligned}&\text{avgprec}(f) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|y_i|} \\ &\sum_{y\in y_i}\frac{|\{y|rank_f(\mathbf{x}_i, y') \leq rank_f(\mathbf{x}_i, y), y' \in y_i\}|}{rank_f(\mathbf{x}_i, y)}.\end{aligned} \tag{13}$$

   This metric has an optimal value of 1.

4.1.2. Label-based metrics

For label-based metrics, we first must define the four basic quantities that characterize the binary classification of each label based on function $h(\cdot)$:

$$
\begin{aligned}
TP_j &= |\{\mathbf{x}_i|y_j \in y_i \wedge y_j \in h(\mathbf{x}_i), 1 \leq i \leq p\}|, \\
FP_j &= |\{\mathbf{x}_i|y_j \notin y_i \wedge y_j \in h(\mathbf{x}_i), 1 \leq i \leq p\}|, \\
TN_j &= |\{\mathbf{x}_i|y_j \notin y_i \wedge y_j \notin h(\mathbf{x}_i), 1 \leq i \leq p\}|, \\
FN_j &= |\{\mathbf{x}_i|y_j \in y_i \wedge y_j \notin h(\mathbf{x}_i), 1 \leq i \leq p\}|.
\end{aligned}
$$

With these four quantities, most of the binary classification metrics can be adapted to multilabel problems. Considering any measure $B(TP_j, FP_j, TN_j, FN_j)$, the label-based classification metric can be obtained either by microaveraging or macroaveraging [44]:

- Macroaveraging:

$$B_{\text{macro}} = \frac{1}{q}\sum_{j=1}^{q} B(TP_j, FP_j, TN_j, FN_j). \tag{14}$$

- Microaveraging:

$$B_{\text{micro}} = B\left(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN_j\right). \tag{15}$$

With these definitions, we can obtain the macro- and microaveraged versions of the accuracy, recall, precision and F^{β} metrics. It is evident that the accuracy metric is the same when macro- and microaveraged.

As in the previous case of example-based metrics, if the intermediate function $f(\cdot)$ is known, we can define two ranking metrics. The first metric is the macroaveraged AUC:

$$
\begin{gathered}
AUC_{\text{macro}} = \frac{1}{q}\sum_{j=1}^{q} AUC_j \\
= \tfrac{1}{q}\textstyle\sum_{j=1}^{q} \frac{|\{(\mathbf{x}',\mathbf{x}'')\}|f(\mathbf{x})',y_j)\geq f(\mathbf{x}'',y_j),(\mathbf{x}',\mathbf{x}'')\in \mathcal{Z}_j\times\tilde{\mathcal{Z}}_j|}{|\mathcal{Z}_j||\tilde{\mathcal{Z}}_j|},
\end{gathered}
\tag{16}
$$

where $\mathcal{Z}_j = \{\mathbf{x}_i|y_j \in Y_i, 1 \leq i \leq p\}$ ($\tilde{\mathcal{Z}}_j = \{\mathbf{x}_i|y_j \in Y_i, 1 \leq i \leq p\}$) is the set of instances with (without) label y_i in its set of relevant labels. The corresponding microaveraged AUC is given by:

$$AUC_{\text{micro}} = \frac{|\{(\mathbf{x}',\mathbf{x}'',y',y'')|f(\mathbf{x}',y')\geq f(\mathbf{x}'',y''),(\mathbf{x}',y')\in S^{+},(\mathbf{x}'',y'')\in S^{-}|}{|S^{+}||S^{-}|\}}, \quad (17)$$

where $\mathcal{S}^{+} = \{(\mathbf{x}_i, y)|y \in Y_i, 1 \leq i \leq p\}$ ($\mathcal{S}^{-} = \{(\mathbf{x}_i, y)|y \notin Y_i, 1 \leq i \leq p\}$) corresponds to the set of relevant (irrelevant) instance-label pairs.

In the experiments, we show the results using all of these metrics or a subset of them for certain cases when studying the behavior of our proposal. The use of different metrics is justified because they represent the performance of the models from different points of view.

5. Experimental results

The first set of experiments was devoted to comparing our proposal against the previously published methods. One of the problems faced is the lack of previous approaches addressing the problem of instance selection for multilabel problems. However, for single label datasets, genetic algorithms have been proven to be the best approach [45]; thus, as the basic method, we have chosen to compare our cooperative approach with a CHC genetic algorithm. To allow a fair comparison, we made the two methods as close as possible in terms of hyperparameters. The same fitness function was used for the cooperative approach (see Eq. 1) and the standard evolutionary algorithm. Additionally, we also compared our method with the ML-kNN algorithm without instance selection. Table 5 shows the parameters used for evolving the cooperative algorithm. The standard evolutionary algorithm and the cooperative algorithm used a population of 100 individuals and were evolved for an hour.

Both algorithms have been run 19 times and the metrics shown above were calculated as the classification performance measures (seq Eq. 1). The results have been divided into different tables for better readability. We have used two tables for the classification-based metrics, one for the example-based metrics and another for the label-based metrics. Another table has been used for ranking measures. The tables showing classification performance results also show the results of an ML-kNN algorithm without instance selection, and $k = 10$ which is the value recommended by the authors [3].

Tables 3, 4, 5 and 6 show the comparisons in terms of reduction using example-based classification metrics, label-based classification metrics and

Table 2: Parameters used for the cooperative coevolutionary algorithm

Parameters	Value
Size of global population (N_g)	100
Size of subpopulations (N_s)	100
Number of generations (G)	1000
k	$[1, 100]$
Internal population iteration (N_p)	10
Internal subpopulation iteration (N_{sp})	10
Memetic mutation (p_m)	0.01
Random bit mutation (p_{rm})	0.01
Bit mutation (p_{bit})	0.1
Drop3 mutation (p_{drop3})	0.01
OSS mutation (p_{oss})	0.01
Elitism (p_{elite})	0.05

ranking metrics, respectively for the fitness function. The comparisons in terms of the classification performance metrics are shown in Tables 7, 8, 9. 10, 11 and 12 for the example-based classification metrics, the label-based classification metrics and the ranking metrics, respectively. Tables 1-19 of the supplementary material show detailed results for reduction and Tables 20-38 of the supplementary material show the detailed results for classification performance. Regarding the reduction, the tables show that the proposed approach achieved a better reduction than the CHC for all 19 different metrics. In all cases, the differences were significant according to the Wilcoxon test. Furthermore, the differences were large, and in most cases they favored our method by approximately 10%.

Regarding the classification performance, our algorithm was also better than CHC. Regarding the classification example-based metrics, our proposal beat the CHC algorithm at a confidence level of 95% in three out of six metrics, including the subset accuracy, precision and recall; and it was better but not significantly different for the accuracy and F1. Our method was significantly worse than CHC only on the Hamming loss. The comparison to the ML-kNN always favored our method, showing the usefulness of instance selection at both reducing the dataset and improving the classification performance of the ML-kNN. Regarding the label-based classification metrics, our method was better than CHC in four metrics, including the recall-macro, F1-macro, recall-micro and F1-micro; worse in two metrics, including the

Table 3: Comparison between our approach and a CHC genetic algorithm in terms of reduction for example-based classification performance metrics as the fitness function

		CHC	Cooperative
Mean		0.8111	0.9499
Ranks		1.9357	1.0643
CHC	w/l		65/4
	p		0.0000
	R^+/R^-		2438.5/46.5

(a) Subset accuracy

		CHC	Cooperative
Mean		0.9360	0.9934
Ranks		1.8571	1.1429
CHC	w/l		57/7
	p		0.0000
	R^+/R^-		2377.0/108.0

(b) Hamming loss

		CHC	Cooperative
Mean		0.8374	0.9468
Ranks		1.9143	1.0857
CHC	w/l		64/6
	p		0.0000
	R^+/R^-		2438.5/46.5

(c) Accuracy (example-based)

		CHC	Cooperative
Mean		0.8477	0.9472
Ranks		1.9857	1.0143
CHC	w/l		69/1
	p		0.0000
	R^+/R^-		2483.0/2.0

(d) Precision (example-based)

		CHC	Cooperative
Mean		0.8453	0.9748
Ranks		1.9214	1.0786
CHC	w/l		64/5
	p		0.0000
	R^+/R^-		2449.5/35.5

(e) Recall (example-based)

		CHC	Cooperative
Mean		0.8430	0.9212
Ranks		1.7786	1.2214
CHC	w/l		54/15
	p		0.0000
	R^+/R^-		2093.5/391.5

(f) F1 (example-based)

Table 4: Comparison between our approach and a CHC genetic algorithm in terms of reduction for the label-based classification performance metrics as the fitness function

		CHC	Cooperative
Mean		0.9341	0.9926
Ranks		1.8857	1.1143
	w/l		59/5
CHC	p		0.0000
	R^+/R^-		2392.5/92.5

(a) Accuracy macro/micro-averaged

		CHC	Cooperative
Mean		0.8433	0.9379
Ranks		1.9000	1.1000
	w/l		62/6
CHC	p		0.0000
	R^+/R^-		2391.5/93.5

(b) Precision (macro-averaged)

		CHC	Cooperative
Mean		0.8303	0.9760
Ranks		1.8857	1.1143
	w/l		60/6
CHC	p		0.0000
	R^+/R^-		2399.0/86.0

(c) Recall (macro-averaged)

		CHC	Cooperative
Mean		0.7987	0.8466
Ranks		1.6714	1.3286
	w/l		47/23
CHC	p		0.0001
	R^+/R^-		1927.0/558.0

(d) F1 (macro-averaged)

accuracy-macro and precision-micro; and there were no differences for the precision-macro. As it was the case for the example-based metrics, the cooperative method improved the results compared to the ML-kNN for all instances for all metrics with the exception of the precision-macro.

Regarding the ranking metrics (see Tables 11 and 12), there were no significant differences for the one-error, coverage, average precision and AUC-micro. Regarding the ranking loss and AUC-macro, our proposal performed better than the standard algorithm. We must stress that this good performance was achieved while very significantly improving the reduction ability of the algorithm. The comparison to the ML-kNN with all instances favored

Table 5: Comparison between our approach and a CHC genetic algorithm in terms of reduction for the label-based classification performance metrics as the fitness function (continuation).

		CHC	Cooperative
Mean		0.8575	0.9657
Ranks		1.9429	1.0571
	w/l		65/3
CHC	p		0.0000
	R^+/R^-		2458.5/26.5

(e) Precision (micro-averaged)

		CHC	Cooperative
Mean		0.8597	0.9870
Ranks		1.9000	1.1000
	w/l		61/5
CHC	p		0.0000
	R^+/R^-		2434.0/51.0

(f) Recall (micro-averaged)

		CHC	Cooperative
Mean		0.8351	0.9367
Ranks		1.8929	1.1071
	w/l		61/6
CHC	p		0.0000
	R^+/R^-		2411.0/74.0

(g) F1 (micro-averaged)

our method for the six different metrics.

Figures 2, 3 and 4 plot the results for every metric using the average of the reduction and performance. This plot is a way of summarizing both values in a single figure. When the best value for a metric m is the smallest one, we use $1 - m$ in the plot. This means that the points above the main diagonal represent datasets for which our method performed better than CHC. Figure 2 shows the results for example-based classification metrics, Figure 3 shows the results for the label-based classification metrics and Figure 4 shows the results for the ranking metrics. The coverage metric is not plotted since its range is different from the range of the reduction.

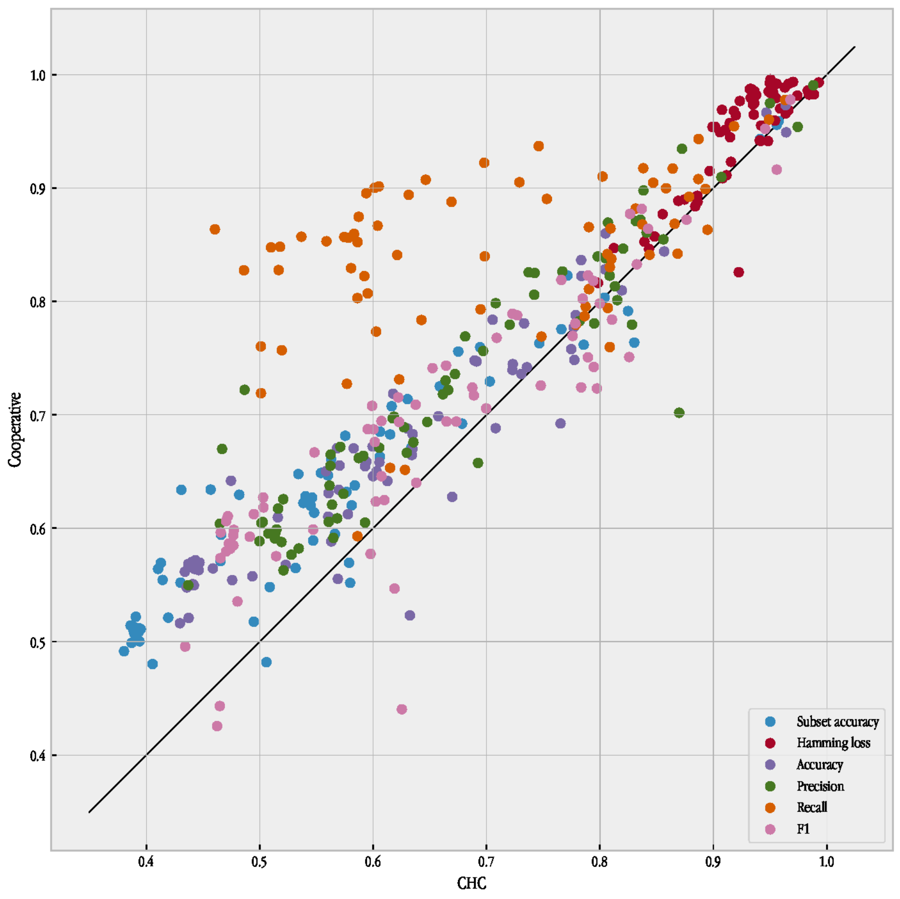

Figure 2: Average values for the reduction and the example-based classification metrics for our cooperative approach and the standard algorithm.

Table 6: Comparison between our approach and a CHC genetic algorithm in terms of reduction for the example- and label-based ranking performance metrics as fitness functions

		CHC	Cooperative
Mean		0.8502	0.9493
Ranks		1.9214	1.0786
	w/l		64/5
CHC	p		0.0000
	R^+/R^-		2439.5/45.5

(a) One-error

		CHC	Cooperative
Mean		0.9209	0.9808
Ranks		1.9143	1.0857
	w/l		60/2
CHC	p		0.0000
	R^+/R^-		2412.0/73.0

(b) Coverage

		CHC	Cooperative
Mean		0.9223	0.9863
Ranks		1.8643	1.1357
	w/l		57/6
CHC	p		0.0000
	R^+/R^-		2365.0/120.0

(c) Ranking-loss

		CHC	Cooperative
Mean		0.8770	0.9520
Ranks		1.9357	1.0643
	w/l		65/4
CHC	p		0.0000
	R^+/R^-		2458.5/26.5

(d) Average precision

		CHC	Cooperative
Mean		0.9077	0.9990
Ranks		1.8214	1.1786
	w/l		45/0
CHC	p		0.0000
	R^+/R^-		2322.5/162.5

(e) AUC macro-averaged

		CHC	Cooperative
Mean		0.8427	0.9354
Ranks		1.8429	1.1571
	w/l		58/10
CHC	p		0.0000
	R^+/R^-		2205.0/280.0

(f) AUC micro-averaged

Table 7: Comparison between our approach, the MLkNN and a CHC genetic algorithm in terms of the example-based classification performance metrics.

		MLkNN	CHC	Cooperative
Mean		0.1587	0.2883	0.3008
Ranks		2.8714	1.6929	1.4357
MLkNN	w/l		64/3	63/2
	p		0.0000	0.0000
	R^+/R^-		2377.0/108.0	2448.5/36.5
CHC	w/l			41/23
	p			0.0473
	R^+/R^-			1581.5/903.5

(a) Subset accuracy

		MLkNN	CHC	Cooperative
Mean		0.1114	0.0803	0.0873
Ranks		2.6000	1.4643	1.9357
MLkNN	w/l		60/10	52/18
	p		0.0000	0.0000
	R^+/R^-		2272.0/213.0	2042.0/443.0
CHC	w/l			20/45
	p			0.0017
	R^+/R^-			707.5/1777.5

(b) Hamming loss

		MLkNN	CHC	Cooperative
Mean		0.2944	0.3916	0.3938
Ranks		2.7571	1.8143	1.4286
MLkNN	w/l		59/11	64/6
	p		0.0000	0.0000
	R^+/R^-		2289.0/196.0	2355.0/130.0
CHC	w/l			46/24
	p			0.3387
	R^+/R^-			1406.0/1079.0

(c) Accuracy (example-based)

Table 8: Comparison between our approach, the MLkNN and a CHC genetic algorithm in terms of the example-based classification performance metrics (continuation).

		MLkNN	CHC	Cooperative
Mean		0.3247	0.4887	0.4996
Ranks		2.9000	1.6714	1.4286
MLkNN	w/l		65/5	68/2
	p		0.0000	0.0000
	R^+/R^-		2394.0/91.0	2463.0/22.0
CHC	w/l			41/27
	p			0.0215
	R^+/R^-			1635.5/849.5

(d) Precision (example-based)

		MLkNN	CHC	Cooperative
Mean		0.4539	0.5603	0.7111
Ranks		2.6857	1.9071	1.4071
MLkNN	w/l		59/11	59/11
	p		0.0000	0.0000
	R^+/R^-		2225.0/260.0	2370.0/115.0
CHC	w/l			52/17
	p			0.0000
	R^+/R^-			2100.5/384.5

(e) Recall (example-based)

		MLkNN	CHC	Cooperative
Mean		0.3687	0.4515	0.4586
Ranks		2.5571	1.8357	1.6071
MLkNN	w/l		52/18	57/13
	p		0.0000	0.0000
	R^+/R^-		2167.0/318.0	2248.0/237.0
CHC	w/l			40/29
	p			0.1550
	R^+/R^-			1485.5/999.5

(f) F1 (example-based)

Table 9: Comparison between our approach, the MLkNN and a CHC genetic algorithm in terms of the label-based classification performance metrics.

		MLkNN	CHC	Cooperative
Mean		0.8886	0.9188	0.9148
Ranks		2.6714	1.4571	1.8714
MLkNN	w/l		61/9	56/14
	p		0.0000	0.0000
	R^+/R^-		2246.0/239.0	2151.0/334.0
CHC	w/l			20/44
	p			0.0016
	R^+/R^-			704.5/1780.5

(a) Accuracy macro/micro-averaged

		MLkNN	CHC	Cooperative
Mean		0.2822	0.2813	0.2692
Ranks		1.9429	2.0000	2.0571
MLkNN	w/l		32/38	34/36
	p		0.6045	0.6628
	R^+/R^-		1331.0/1154.0	1168.0/1317.0
CHC	w/l			31/37
	p			0.5312
	R^+/R^-			1135.5/1349.5

(b) Precision (macro-averaged)

		MLkNN	CHC	Cooperative
Mean		0.2914	0.3404	0.3932
Ranks		2.4143	2.0071	1.5786
MLkNN	w/l		40/30	59/11
	p		0.0476	0.0000
	R^+/R^-		1581.0/904.0	2177.0/308.0
CHC	w/l			40/29
	p			0.0013
	R^+/R^-			1790.5/694.5

(c) Recall (macro-averaged)

		MLkNN	CHC	Cooperative
Mean		0.2461	0.2702	0.2858
Ranks		2.1286	2.0929	1.7786
MLkNN	w/l		37/33	42/28
	p		0.0375	0.0011
	R^+/R^-		1598.0/887.0	1801.0/684.0
CHC	w/l			43/26
	p			0.0225
	R^+/R^-			1632.5/852.5

(d) F1 (macro-averaged)

Table 10: Comparison between our approach, the MLkNN and a CHC genetic algorithm in terms of the label-based classification performance metrics (continuation).

		MLkNN	CHC	Cooperative
Mean		0.3535	0.5711	0.4558
Ranks		2.6571	1.4429	1.9000
MLkNN	w/l		65/5	51/19
	p		0.0000	0.0009
	R^+/R^-		2260.0/225.0	1808.0/677.0
CHC	w/l			23/41
	p			0.0006
	R^+/R^-			657.5/1827.5

(e) Precision (micro-averaged)

		MLkNN	CHC	Cooperative
Mean		0.4417	0.5395	0.7114
Ranks		2.6786	1.9571	1.3643
MLkNN	w/l		54/16	63/6
	p		0.0000	0.0000
	R^+/R^-		2104.0/381.0	2377.5/107.5
CHC	w/l			49/17
	p			0.0000
	R^+/R^-			2156.5/328.5

(f) Recall (micro-averaged)

		MLkNN	CHC	Cooperative
Mean		0.3738	0.4209	0.4525
Ranks		2.5714	1.9929	1.4357
MLkNN	w/l		48/22	62/8
	p		0.0015	0.0000
	R^+/R^-		1784.0/701.0	2305.0/180.0
CHC	w/l			47/22
	p			0.0008
	R^+/R^-			1814.5/670.5

(g) F1 (micro-averaged)

Table 11: Comparison between our approach, the MLkNN and a CHC genetic algorithm in terms of the example- and label-based ranking metrics.

		MLkNN	CHC	Cooperative
Mean		0.6131	0.4848	0.5182
Ranks		2.5714	1.8071	1.6214
MLkNN	w/l		54/13	54/15
	p		0.0000	0.0000
	R^+/R^-		2227.0/258.0	1969.5/515.5
CHC	w/l			39/25
	p			0.3476
	R^+/R^-			1403.0/1082.0

(a) One-error

		MLkNN	CHC	Cooperative
Mean		29.5161	23.8921	23.7328
Ranks		2.5500	1.6429	1.8071
MLkNN	w/l		55/14	53/17
	p		0.0000	0.0002
	R^+/R^-		2148.5/336.5	1883.5/601.5
CHC	w/l			28/37
	p			0.2104
	R^+/R^-			1028.5/1456.5

(b) Coverage

		MLkNN	CHC	Cooperative
Mean		0.1481	0.0447	0.0359
Ranks		2.8571	1.7429	1.4000
MLkNN	w/l		65/3	63/5
	p		0.0000	0.0000
	R^+/R^-		2429.5/55.5	2404.5/80.5
CHC	w/l			43/17
	p			0.0036
	R^+/R^-			1740.5/744.5

(c) Ranking-loss

Table 12: Comparison between our approach, the MLkNN and a CHC genetic algorithm in terms of the example- and label-based ranking metrics.

		MLkNN	CHC	Cooperative
Mean		0.4698	0.5499	0.5513
Ranks		2.6143	1.7714	1.6143
MLkNN	w/l		59/11	54/16
	p		0.0000	0.0000
	R^+/R^-		2184.0/301.0	2122.0/363.0
CHC	w/l			43/27
	p			0.1646
	R^+/R^-			1480.0/1005.0

(d) Average precision

		MLkNN	CHC	Cooperative
Mean		0.8290	0.9024	0.9110
Ranks		2.9714	1.8286	1.2000
MLkNN	w/l		68/1	69/0
	p		0.0000	0.0000
	R^+/R^-		2482.5/2.5	2484.5/0.5
CHC	w/l			43/0
	p			0.0000
	R^+/R^-			2296.0/189.0

(e) AUC macro-averaged

		MLkNN	CHC	Cooperative
Mean		0.8011	0.8457	0.8381
Ranks		2.6071	1.6214	1.7714
MLkNN	w/l		59/11	53/16
	p		0.0000	0.0000
	R^+/R^-		2273.0/212.0	2134.5/350.5
CHC	w/l			32/37
	p			0.0848
	R^+/R^-			948.0/1537.0

(f) AUC micro-averaged

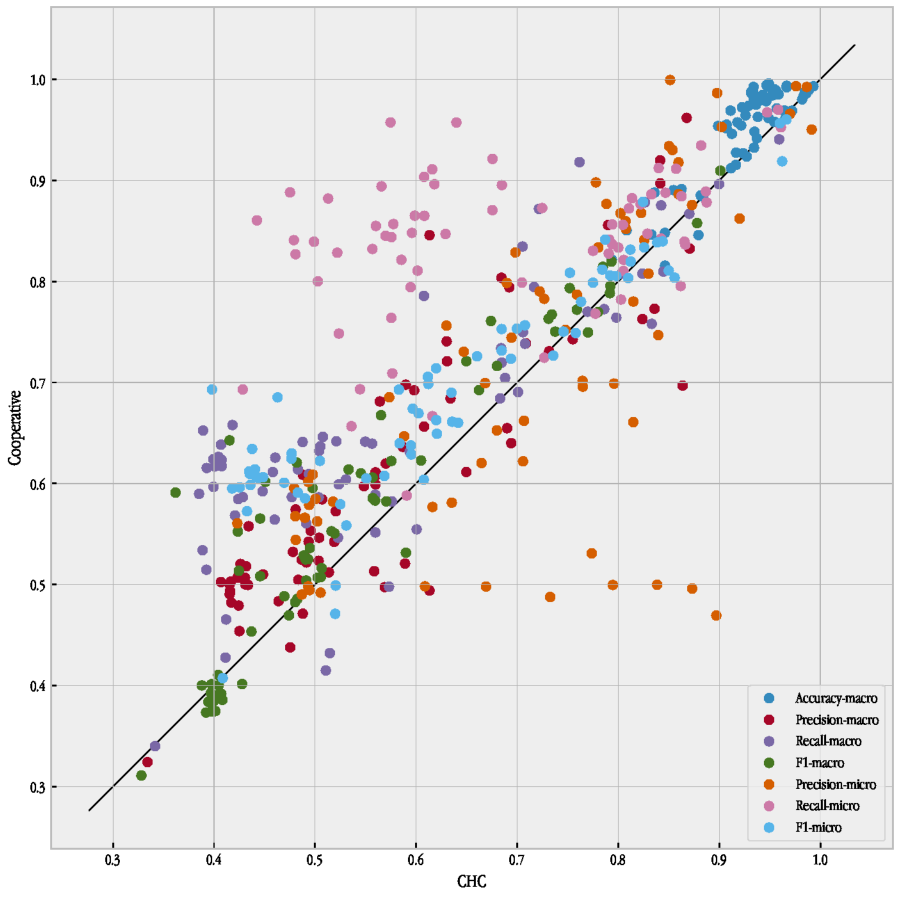

Figure 3: Average values for the reduction and the label-based classification metrics for our cooperative approach and the standard algorithm.

The three figures show similar results. For a large majority of the datasets, the combined value of the classification performance and reduction is better for our proposal than the standard CHC evolutionary algorithm. The only overall performance metric for which our proposal is worse is the precision-macro.

5.1. Combination of best the subindividuals

Most of the previous applications of cooperative coevolution to different problems did not use a population of combinations [6]. Instead, the best individual of every subpopulation is used or a certain prearranged combination

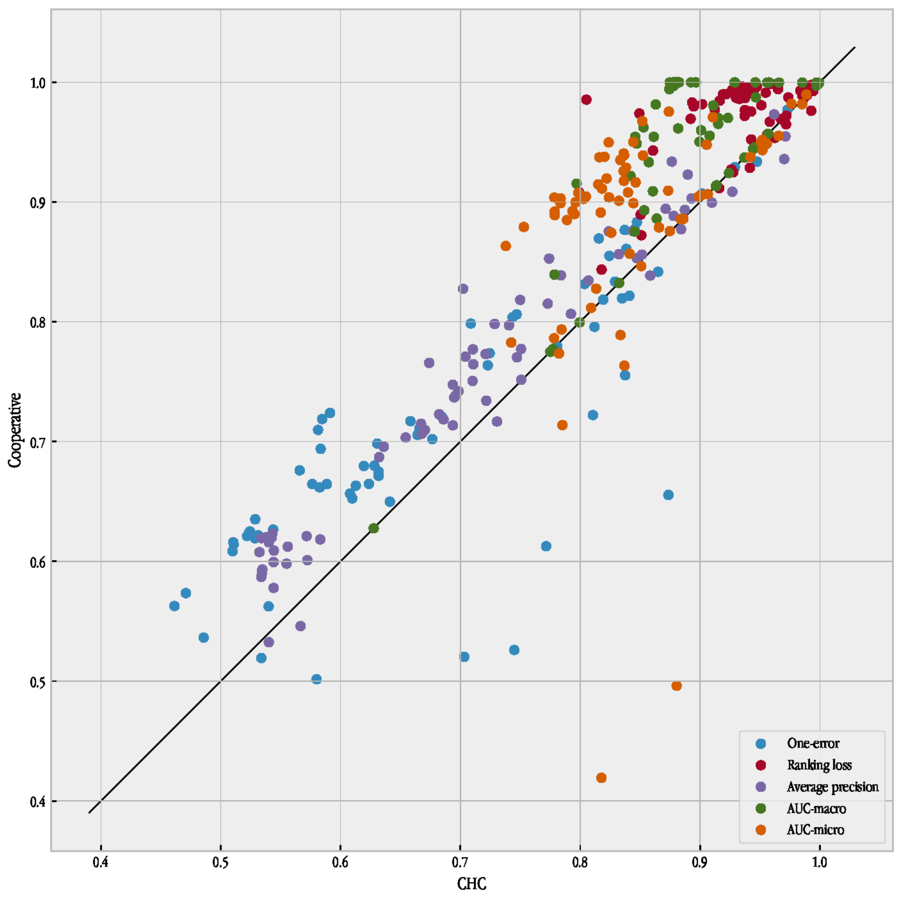

Figure 4: Average values for the reduction and the ranking metrics for our cooperative approach and the standard algorithm.

is developed based on the decomposition of the problem. Since our approach is different, we must validate the use of a separate population for evolving the best combinations of subindividuals. We conducted experiments without using the global population of combinations and using the combination of the best individual of every subpopulation as the solution to the instance selection problem. These experiments were carried out for ten classification metrics selected as representative of the 19 used above: the subset accuracy, Hamming loss, accuracy, F1, recall, one-error, coverage, ranking loss, AUC and average precision.

Figure 5 shows the comparison using a radar plot in terms of both classification performance and reduction. For coverage, since the scale is different, we plot the average of $\frac{\text{Optimal coverage}}{\text{Coverage}}$ whose optimum value is 1. For both the reduction and classification performance, the results supported the use of the global population. Almost all metrics showed worse performance when the population of combinations was removed. Notably, the reduction ability was clearly damaged. It seems evident from the results that combining the best individuals is a suboptimal solution when compared with the use of a population of combinations evolved together with the subpopulations.

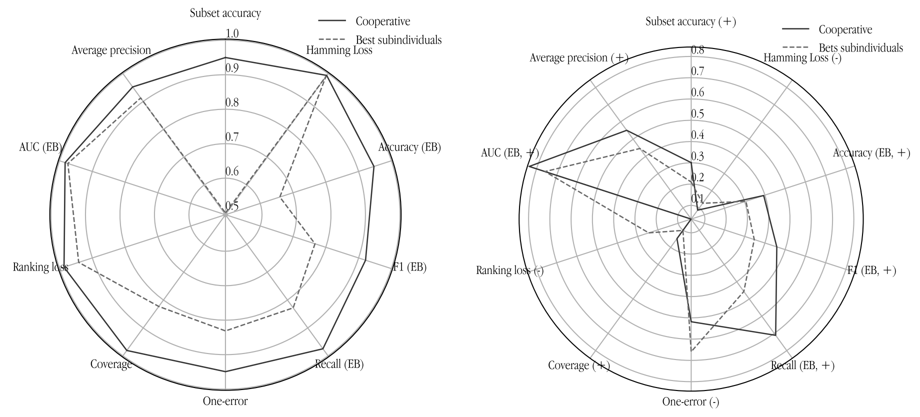

Figure 5: Average values for the reduction (left) and performance (right) for the 10 performance metrics selected for our cooperative approach and the same method removing the global population.

5.2. Removing local search

As explained in Section 3, the evolution of the subpopulations includes a local search step implemented as a mutation operator. This part of the algorithm was also validated experimentally. We conducted experiments removing the local search step for the ten performance metrics of the previous section. Figure 6 shows the radar plot for the average values of the reduction and the corresponding performance metric.

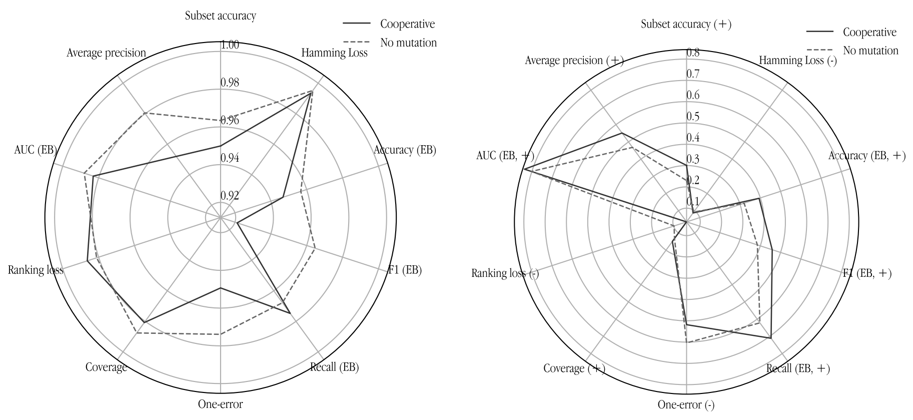

Figure 6: Average values for the reduction (left) and performance (right) for the 10 performance metrics selected for our cooperative approach and the same method removing the local search operators.

The trend shown in figure is different from that of the previous case. The classification performance is less affected by the removal of the local search with lesser differences between the performance with and without local search. However, overall, the method with the local search performed better. Regarding the reduction, the results showed a clear improvement when the local search was removed. This might be an unexpected result since the local search algorithm's main aim is reduction. However, a closer look at the evolution of the algorithm showed a clear pattern. Due to the large class imbalance of some of the labels, the local search achieved a large reduction, removing almost all instances. That produced useless individuals that negatively affected the reduction ability of the method since the resulting global individuals had very poor classification performance and were thus discarded during the evolution.

5.3. Considering the multilabel global performance for subindividuals

Our next experiment consisted of studying the behavior of our model when multilabel global fitness was added for the subindividuals. This term evaluated the fitness of the individual using the same multilabel metric used for the population of combinations and added this term to Eq. 2 for a new fitness value f'_{ij}:

$$f'_{ij} = c_i(s_{ij}) + r_{ij} + \bar{F} + \text{multilabel performance}, \tag{18}$$

The aim was to improve the performance by means of a more global view from the subindividuals. The multilabel performance is always evaluated for $k = 10$ since the subindividuals did not affect the value of k. Figure 7 shows the average values of the reduction and the 10 performance metrics for the standard version of our approach using the previous fitness function for the subindividuals. The plots show that adding the multilabel performance measure improved the reduction ability of the method but damaged its classification performance.

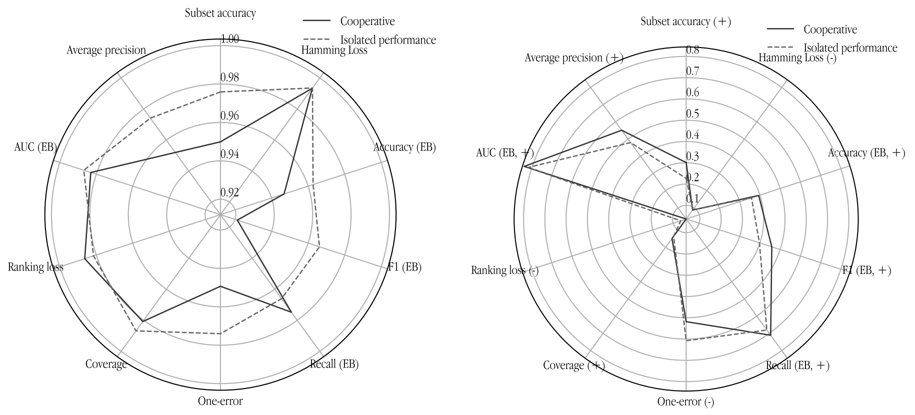

Figure 7: Average values for the reduction (left) and performance (right) for the 10 performance metrics selected for our cooperative approach and the same method using multilabel performance as part of the subindividuals' fitness.

5.4. Using a fixed quorum sensing threshold

Another aspect of our method is that the quorum sensing threshold for retaining an instance is evolved along with the selection of individuals from the subpopulations in the population of combinations. As an additional test of the usefulness of the evolution of this threshold, we conducted experiments for selecting an instance with a fixed threshold of 10%. Figure 8 shows the radar plot comparing the evolutionary and fixed values for the quorum sensing threshold.

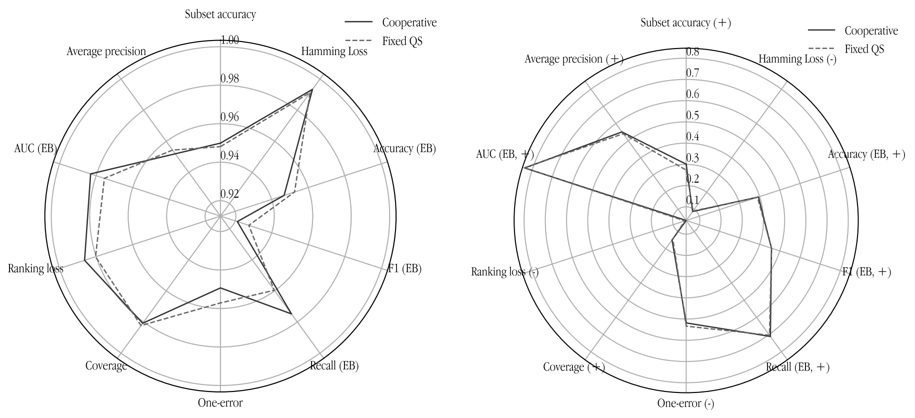

Figure 8: Average values for the reduction (left) and performance (right) for the 10 performance metrics selected for our cooperative approach and the same method using a fixed quorum sensing threshold.

In this case, the differences were small. Both versions of the algorithms achieved almost the same results in terms of both reduction and classification performance. Since the evolution of the subindividuals also depended on the threshold due to the combination fitness, the subindividuals learned to adapt to the fixed threshold and the dynamic one.

5.5. Class-imbalanced dataset approach

As has been previously stated, when considered alone, many of the labels presented a heavily class imbalance classification problem. This is especially relevant for datasets with large degrees of label sparsity. To account for the fact that we designed a version of our method taking into account the class imbalance nature of the problems faced by each subpopulation, in this version, prior to evolution, we applied undersampling for every subpopulation, taking into account its corresponding label. Therefore, subpopulation i was evolved using an undersampled sample of the whole dataset using its corresponding label. This sampling meant that every subpopulation was evolved with different, though overlapped, datasets. The remainder of the algorithm was performed using the standard cooperative method described above.

Figure 9 shows the radar plot for the two versions of the algorithms. Regarding the reduction, the class imbalance version achieved better results. Since every subpopulation began with a smaller sample, they are expected to obtain larger reductions. In terms of classification performance, the standard version performed better for all metrics, although the differences were not large. Additionally, the class imbalance version was faster since the subpopulations had to deal with smaller datasets.

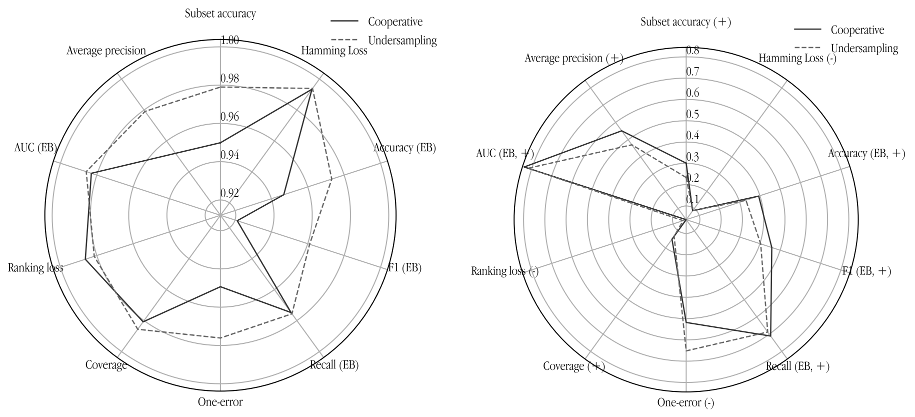

Figure 9: Average values for the reduction (left) and performance (right) for the 10 performance metrics selected for our cooperative approach and the same method adding undersampling.

6. Conclusions

In this paper, we have proposed the first cooperative coevolutionary algorithm for instance selection for multilabel problems. The method is based on two populations that evolve together. One of the populations formed by different subpopulations that evolve individuals focused on a particular label. The other population evolves combinations of the individuals of the subpopulation to obtain valid solutions for the global problem.

We have compared our method with the evolution of a genetic algorithm in the standard way. The results have shown the better performance of our method in terms of the reduction and classification performance using 19 different metrics. Our method also improved the results of an ML-kNN algorithm using all the instances of the datasets. A further study of the different components of our algorithm has validated the design of the method.

There are many different ways that the work presented here can be continued. One of the most straightforward is using a multiobjective approach for the evolution of both populations since different terms are used for the fitness function. Another field where similar ideas could be applied is the feature selection task. Furthermore, the design of ensembles of classifiers for multilabel problems could also benefit from the ideas reported in this paper.

References

[1] M. L. Zhang, Z. H. Zhou, A review on multi-label learning algorithms, IEEE Transactions on Knowledge and Data Engineering 26 (2014) 1819–1837.

[2] J. Read, L. Martino, P. M. Olmos, D. Luengo, Scalable multi-output label prediction: From classifier chains to classifier trellises, Pattern Recognition 48 (2015) 2096–2109.

[3] M.-L. Zhang, Z.-H. Zhou, Ml-knn: A lazy learning approach to multi-label learning, Pattern Recognition 40 (2007) 2038–2048.

[4] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Mining and Knowledge Discovery 6 (2002) 153–172.

[5] M. A. Potter, The design and analysis of a computational model of cooperative coevolution, Ph.D. thesis, George Mason University, Fairfax, Virginia (1997).

[6] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, Z. Zhu, A survey on cooperative co-evolutionary algorithms, IEEE Transactions on Evolutionary Computation 23 (2019) 421–441.

[7] M. A. Potter, K. A. De Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, Evolutionary Computation 8 (1) (2000) 1–29.

[8] R. Cheng, M. N. Omidvar, A. H. Gandomi, B. Sendhoff, S. Menzel, X. Yao, Solving incremental optimization problems via cooperative coevolution, IEEE Transactions on Evolutionary Computation 23 (2019) 762–775.

[9] D. Gong, B. Xu, Y. Zhang, Y. Guo, S. Yang, A similarity-based cooperative co-evolutionary algorithm for dynamic interval multiobjective optimization problems, IEEE Transactions on Evolutionary Computation 24 (2020) 142–156.

[10] N. R. Sabar, J. Abawajy, J. Yearwood, Heterogeneous cooperative coevolution memetic differential evolution algorithm for big data optimization problems, IEEE Transactions on Evolutionary Computation 21 (2017) 315–327.

[11] A. Farahmand, M. N. Ahmadabadi, C. Lucas, B. N. Araabi, Interaction of culture-based learning and cooperative coevolution and its application to automatic behavior-based system desig, IEEE Transactions on Evolutionary Computation 14 (2010) 23–57.

[12] A. Arnaiz-González, J. F. Díez-Pastor, J. J. Rodríguez, C. García-Osorio, Local sets for multi-label instance selection, Applied Soft Computing Journal 68 (2018) 651–666.

[13] M. Kordos, A. Arnaiz-González, C. García-Osorio, Evolutionary prototype selection for multi-output regression, Neurocomputing 358 (2019) 309–320.

[14] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Information Sciences 178 (2008) 2985–2999.

[15] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Transactions on Evolutionary Computation 16 (2012) 210–224.

[16] M. Omidvar, X. Y. Mei, X. Yao, Cooperative coevolution with differential grouping for large scale optimization, IEEE Transactions on Evolutionary Computation 18 (2014) 378–393.

[17] Y.-H. Jia, W.-N. Chen, T. Gu, H. Zhang, H.-Q. Yuan, S. Kwong, J. Zhang, Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization, IEEE Transactions on Evolutionary Computation 23 (2019) 188–202.

[18] K. W. Sun, C. H. Lee, J. Wang, Multilabel classification via coevolutionary multilabel hypernetwork, IEEE Transactions on Knowledge and Data Engineering 28 (2016) 2438–2451.

[19] A. Rosales-Pérez, A. E. Gutiérrez-Rodríguez, J. C. Ortiz-Bayliss, H. Terashima-Marín, C. A. C. Coello, Evolutionary multilabel hyper-heuristic design, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2017, p. 2622–2629.

[20] J. Park, M.-W. Park, D.-W. Kim, J. Lee, Multi-population genetic algorithm for multilabel feature selection based on label complementary communication, Entropy 22 (2020) 876.

[21] N. García-Pedrajas, C. Hervás-Martínez, D. Ortiz-Boyer, Cooperative coevolution of artificial neural network ensembles for pattern classification, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 271–302.

[22] G. A. Trunfio, P. Topa, J. Was, A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution, Information Sciences 372 (2016) 773–795.

[23] X. Lu, S. Menzel, K. Tang, X. Yao, Cooperative coevolution-based design optimization: A concurrent engineering perspective, IEEE Transactions on Evolutionary Computation 22 (2018) 173–188.

[24] R. Chandra, M. Frean, M. Zhang, On the issue of separability for problem decomposition in cooperative neuro-evolution, Neurocomputing 87 (2012) 33–40.

[25] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, COVNET: A cooperative coevolutionary model for evolving artificial neural networks, IEEE Transactions on Neural Networks 14 (3) (2003) 575–596.

[26] F. Zhu, S. U. Guan, Cooperative coevolution of ga-based classifiers based on input decomposition, Engineering Applications in Artificial Intelligence 21 (2008) 1360–1369.

[27] R. P. Wiegand, W. C. Liles, K. A. D. Jong, An empirical analysis of collaboration methods in cooperative coevolutionary algorithms,